Symbolic Logic meets Machine Learning: A Brief Survey in Infinite Domains

Symbolic Logic meets Machine Learning: A Brief Survey in Infinite Domains*

Vaishak Belle

University of Edinburgh & Alan Turing Institute, UK
vaishak@ed.ac.uk

Abstract. The tension between deduction and induction is perhaps the most fundamental issue in areas such as philosophy, cognition and artificial intelligence (AI). The deduction camp concerns itself with questions about the expressiveness of formal languages for capturing knowledge about the world, together with proof systems for reasoning from such knowledge bases. The learning camp attempts to generalize from examples about partial descriptions about the world. In AI, historically, these camps have loosely divided the development of the field, but advances in cross-over areas such as statistical relational learning, neuro-symbolic systems, and high-level control have illustrated that the dichotomy is not very constructive, and perhaps even ill-formed.
In this article, we survey work that provides further evidence for the connections between logic and learning. Our narrative is structured in terms of three strands: logic versus learning, machine learning for logic, and logic for machine learning, but naturally, there is considerable overlap. We place an emphasis on the following "sore" point: there is a common misconception that logic is for discrete properties, whereas probability theory and machine learning, more generally, is for continuous properties. We report on results that challenge this view on the limitations of logic, and expose the role that logic can play for learning in infinite domains.

1 Introduction

The tension between *deduction* and *induction* is perhaps the most fundamental issue in areas such as philosophy, cognition and artificial intelligence (AI). The deduction camp concerns itself with questions about the expressiveness of formal languages for capturing knowledge about the world, together with proof systems for reasoning from such knowledge bases. The learning camp attempts to generalize from examples about partial descriptions about the world. In AI, historically, these camps have loosely divided the development of the field, but advances in cross-over areas such as *statistical relational learning* [83, 38], *neuro-symbolic systems* [37, 28, 60], and *high-level control* [59, 50] have illustrated that the dichotomy is not very constructive, and perhaps even ill-formed. Indeed, logic emphasizes high-level reasoning, and encourages structuring the world in terms of objects, properties, and relations. In contrast, much of the inductive machinery assume random variables to be independent and identically distributed, which can be

* The author was supported by a Royal Society University Research Fellowship. He is grateful to Ionela G. Mocanu, Paulius Dilkas and Kwabena Nuamah for their feedback.

problematic when attempting to exploit symmetries and causal dependencies between groups of objects. But the threads connecting logic and learning go deeper, far beyond the apparent flexibility that logic offers for modeling relations and hierarchies in noisy domains. At a conceptual level, for example, although there is much debate about what precisely commonsense knowledge might look like, it is widely acknowledged that concepts such as time, space, abstraction and causality are essential [68, 98]. In that regard, (classical, or perhaps non-classical) logic can provide the formal machinery to reason about such concepts in a rigorous way. At a pragmatic level, despite the success of methods such as deep learning, it is now increasingly recognized that owing to a number of reasons, including model re-use, transferability, causal understanding, relational abstraction, explainability and data efficiency, those methods need to be further augmented with logical, symbolic and/or programmatic artifacts [17, 97, 35]. Finally, for building intelligent agents, it is recognized that low-level, data-intensive, reactive computations needs to be tightly integrated with high-level, deliberative computations [59, 50, 67], the latter possibly also engaging in hypothetical and counterfactual reasoning. Here, a parallel is often drawn to Kahneman's so-called *System 1* versus *System 2* processing in human cognition [51], in the sense that experiential and reactive processing (learned behavior) needs to be coupled with cogitative processing (reasoning, deliberation and introspection) for sophisticated machine intelligence.

The purpose of this article is not to resolve this debate, but rather provide further evidence for the connections between logic and learning. In particular, our narrative is inspired by a recent symposium on logic and learning [13], where the landscape was structured in terms of three strands:

1. ***Logic vs. Machine Learning****, including the study of problems that can be solved using either logic-based techniques or via machine learning, . . .;*
2. ***Machine Learning for Logic****, including the learning of logical artifacts, such as formulas, logic programs, . . .; and*
3. ***Logic for Machine Learning****, including the role of logics in delineating the boundary between tractable and intractable learning problems, . . . , and the use of logic as a declarative framework for expressing machine learning constructs.*

In this article, we particularly focus on the following "sore" point: there is a common misconception that logic is for discrete properties, whereas probability theory and machine learning, more generally, is for continuous properties. It is true that logical formulas are discrete structures, but they can very easily also express properties about countably infinite or even uncountably many objects. Consequently, in this article we survey some recent results that tackle the integration of logic and learning in infinite domains. In particular, in the context of the above three strands, we report on the following developments. On (1), we discuss approaches for logic-based probabilistic inference in continuous domains. On (2), we cover approaches for learning logic programs in continuous domains, as well as learning formulas that represent countably infinite sets of objects. Finally, on (3), we discuss attempts to use logic as a declarative framework for common tasks in machine learning over discrete and continuous features, as well as using logic as a meta-theory to consider notions such as the *abstraction* of a probabilistic model.

We remark that this survey is undoubtedly a biased view, as the area of research is large, but we do attempt to briefly cover the major threads. Readers are encouraged to refer to discussions in [13, 38, 83], among others, to get a sense of the breadth of the area.

2 Logic vs. Machine Learning

To appreciate the role and impact of logic-based solvers for machine learning systems, it is perhaps useful to consider the core computational problem underlying (probabilistic) machine learning: the problem of inference, including evaluating the partition function (or conditional probabilities) of a probabilistic graphical model such as a Bayesian network.

When leveraging Bayesian networks for machine learning tasks [56], the networks are often learned using local search to maximize a likelihood or a Bayesian quantity. For example, given data $\mathcal{D}$ and the current guess for the network $\mathcal{N}$, we might estimate the "goodness" of the guess by means of a score: $score(\mathcal{N}, \mathcal{D}) \propto \log \Pr(\mathcal{D} \mid \mathcal{N}) - size(\mathcal{N})$. That is, we want to maximize the fit of the data wrt the current guess, but we would like to penalize the model complexity, to avoid overfitting. Then, we would opt for a second guess $\mathcal{N}'$ only if $score(\mathcal{N}', \mathcal{D}) > score(\mathcal{N}, \mathcal{D})$. Needless to say, even with a reasonable local search procedure, the most significant computational effort here is that of probabilistic inference.

Reasoning in such networks becomes especially challenging with logical syntax. The prevalence of large-scale social networks, machine reading domains, and other types of relational knowledge bases has led to numerous formalisms that borrow the syntax of predicate logic for probabilistic modeling [78, 85, 81, 93]. This has led to a large family of solvers for the *weighted model counting* (WMC) problem [39, 20]. The idea is this: given a Bayesian network, a relational Bayesian network, a factor graph, or a probabilistic program [84], one considers an encoding of the formalism as a *weighted propositional theory*, consisting of a propositional theory Δ and a weight function w that maps atoms in Δ to $\mathbb{R}^+$. Recall that SAT is the problem of finding an assignment to such a Δ, whereas #SAT counts the number of assignments for Δ. WMC extends #SAT by computing the sum of the weights of all assignments: that is, given a set of models $\mathcal{M}(\Delta) = \{M \mid M \models \Delta\}$, we evaluate the quantity $W(\Delta) = \sum_{M \in \mathcal{M}(\Delta)} w(M)$ where $w(M)$ is factorized in terms of the atoms true at M. To obtain the conditional probability of a query q against evidence e (wrt the theory Δ), we define $\Pr(q \mid e) = W(\Delta \wedge q \wedge e)/W(\Delta \wedge e)$.

The popularity of WMC can be explained as follows. Its formulation elegantly decouples the logical or symbolic representation from the numeric representation, which is encapsulated in the weight function. When building solvers, this allows us to reason about logical equivalence and reuse SAT solving technology (such as constraint propagation and clause learning). WMC also makes it more natural to reason about deterministic, hard constraints in a probabilistic context [20]. Both exact solvers, based on knowledge compilation [23], as well as approximate solvers [19] have emerged in the recent years, as have lifted techniques [95] that exploit the relational syntax during inference (but in a finite domain setting). For ideas on generating such representations randomly to assess scalability and compare inference algorithms, see [29], for example.

On the point of modelling finite vs infinite properties, note that owing to the underlying propositional language, the formulation is limited to discrete random variables. A similar observation can be made for SAT, which for the longest time could only be applied in discrete domains. This changed with the increasing popularity of *satisfiability modulo theories* (SMT) [4], which enable us to, for example, reason about the satisfiability of linear constraints over the rationals. Extending earlier insights on piecewise-polynomial weight functions [89, 88], the formulation of *weighted model integration* (WMI) was proposed in [12]. WMI extends WMC by leveraging the idea that SMT theories can represent mixtures of Boolean and continuous variables: for example, a formula such as $p \wedge (x > 5)$ denotes the logical conjunction of a Boolean variable p and a real-valued variable x taking values greater than 5. For every assignment to the Boolean and continuous variables, the WMI problem defines a weight. The total WMI is computed by integrating these weights over the domain of solutions to Δ, which is a mixed discrete-continuous (or simply *hybrid*) space. Consider, for example, the special case when Δ has no Boolean variables, and the weight of every model is 1. Then, the WMI simplifies to computing the volume of the polytope encoded in Δ. When we additionally allow for Boolean variables in Δ, this special case becomes the hybrid version of #SAT, known as #SMT [21]. Since that proposal, numerous advances have been made on building efficient WMI solvers (e.g., [74, 69, 99]) including the development of compilation targets [53, 54, 100].

Note that WMI proposes an extension of WMC for uncountably infinite (i.e., continuous) domains. What about countably infinite domains? The latter type is particularly useful for reasoning in (general) first-order settings, where we may say that a property such as $\forall x, y, z(\mathit{parent}(x, y) \wedge \mathit{parent}(y, z) \supset \mathit{grandparent}(x, z))$ applies to every possible x, y and z. Of course, in the absence of the finite domain assumption, reasoning in the first-order setting suffers from undecidability properties, and so various strategies have emerged for reasoning about an *open universe* [87]. One popular approach is to perform *forward reasoning*, where samples needed for probability estimation are obtained from the facts and declarations in the probabilistic model [87, 45]. Each such sample corresponds to a possible world. But there may be (countably or uncountably) infinitely many worlds, and so exact inference is usually sacrificed. A second approach is to restrict the model wrt the query and evidence atoms and define estimation from the resulting finite sub-model [70, 90, 41], which may also be substantiated with exact inference in special cases [6, 7].

Given the successes of logic-based solvers for inference and probability estimation, one might wonder whether such solvers would also be applicable to learning tasks in models with relational features and hard, deterministic constraints? These, in addition to other topics, are considered in the next section.

3 Machine Learning for Logic

At least since the time of Socrates, inductive reasoning has been a core issue for the logical worldview, as we need a mechanism for obtaining axiomatic knowledge. In that regard, the learning of logical and symbolic artifacts is an important issue in AI, and computer science more generally [43]. There is a considerable body of work on learning

propositional and relational formulas, and in context of probabilistic information, learning weighted formulas [13, 75, 26, 83]. Approaches can be broadly lumped together as follows.

1. *Entailment-based scoring:* Given a logical language $\mathcal{L}$, background knowledge $\mathcal{B} \subset \mathcal{L}$, examples $\mathcal{D}$ (usually a set of $\mathcal{L}$-atoms), find a hypothesis $\mathcal{H} \in \overline{\mathcal{H}}, \mathcal{H} \subset \mathcal{L}$ such that $\mathcal{B} \cup \mathcal{H}$ entail the instances in $\mathcal{D}$. Here, the set $\overline{\mathcal{H}}$ places restrictions of the syntax of $\mathcal{H}$ so as to control model complexity and generalization. (For example, $\mathcal{H} = \mathcal{D}$ is a trivial hypothesis that satisfies the entailment stipulation.)
2. *Likelihood-based scoring:* Given $\mathcal{L}, \mathcal{B}$ and $\mathcal{D}$ as defined above, find $\mathcal{H} \subset \mathcal{L}$ such that $\mathit{score}(\mathcal{H}, \mathcal{D}) > \mathit{score}(\mathcal{H}', \mathcal{D})$ for every $\mathcal{H}' \neq \mathcal{H}$. As discussed before, we might define $\mathit{score}(\mathcal{H}, \mathcal{D}) \propto \log \Pr(\mathcal{D} \mid \mathcal{H}) - \mathit{size}(\mathcal{H})$. Here, like $\overline{\mathcal{H}}$ above, $\mathit{size}(\mathcal{H})$ attempts to the control model complexity and generalization.

Many recipes based on these schemes are possible. For example, we may use entailment-based inductive synthesis for an initial estimate of the hypothesis, and then resort to Bayesian scoring models [85]. The synthesis step might invoke neural machinery [35]. We might not require that the hypothesis entails every example in $\mathcal{D}$ but only the largest consistent subset, which is sensible when we expect the examples to be noisy [26]. We might compile $\mathcal{B}$ to an efficient data structure, and perform likelihood-based scoring on that structure [63], and so $\mathcal{B}$ could be seen as deterministic domain-specific constraints. Finally, we might stipulate the conditions under which a "correct" hypothesis may be inferred wrt unknown ground truth, only a subset of which is provided in $\mathcal{D}$. This is perhaps best represented by the (probably approximately correct) PAC-semantics that captures the quality possessed by the output of learning algorithm whilst costing for the number of examples that need to be observed [94, 22]. (But other formulations are also possible, e.g., [42].)

This discussion pertained to finite domains. What about continuous spaces? By means of arithmetic fragments and formulations like WMI, it should be clear that it now becomes possible to extend the above schemes to learn continuous properties. For example, one could learn linear expressions from data [55]. For an account that also tries to evaluate a hypothesis that is correct wrt unknown ground truth, see [72]. If the overall objective is to obtain a distribution of the data, other possibilities present themselves. In [77], for example, real-valued data points are first lumped together to obtain atomic continuous random variables. From these, relational formulas are constructed so as to yield hybrid probabilistic programs. The learning is based on likelihood scoring. In [91], the real-valued data points are first intervalized, and polynomials are learned for those intervals based on likelihood scoring. These weighted atoms are then used for learning clauses by entailment judgements [26].

Such ideas can also be extended to data structures inspired by knowledge compilation, often referred to as *circuits* [20, 82]. Knowledge compilation [25] arose as a way to represent logical theories in a manner where certain kinds of computations (e.g., checking satisfiability) is significantly more effective, often polynomial in the size of the circuit. In the context of probabilistic inference, the idea was to then position probability estimation to also be computable in time polynomial in the size of the circuit [20, 82]. Consequently, (say) by means of likelihood-based scoring, the learning of circuits

is particularly attractive because once learned, the bottleneck of inference is alleviated [66, 63]. In [73, 15], along the lines of the work above on learning logical formulas in continuous domains, it is shown that the learning of circuits can also be coupled with WMI.

What about countably infinite domains? In most pragmatic instances of learning logical artifacts, the difference between the uncountable and countably infinite setting is this: in the former, we see finitely many real-valued samples as being drawn from an (unknown) interval, and we could inspect these samples to crudely infer a lower and upper bound. In the latter, based on finitely many relational atoms, we would need to infer a universally quantified clause, such as $\forall x, y, z(parent(x, y) \wedge parent(y, z) \supset grandparent(x, z))$. If we are after a hypothesis that is simply guaranteed to be consistent wrt the observed examples, then standard rule induction strategies would suffice [75], and we could interpret the rules as quantifying over a countably infinite domain. But this is somewhat unsatisfactory, as there is no distinction between the rules learned in the standard finite setting and its supposed applicability to the infinite setting. What is really needed is an analysis of what rule learning would mean wrt the infinitely many examples that have *not* been observed. This was recently considered via the PAC-semantics in [10], by appealing to ideas on reasoning with open universes discussed earlier [6].

Before concluding this section, it is worth noting that although the above discussion is primarily related to the learning of logical artifacts, it can equivalently be seen as a class of machine learning methods that leverage symbolic domain knowledge [30]. Indeed, logic-based probabilistic inference over deterministic constraints, and entailment-based induction augmented with background knowledge are instances of such a class. Analogously, the automated construction of relational and statistical knowledge bases [79, 18] by combining background knowledge with extracted tuples (obtained, for example, by applying natural language processing techniques to large textual data) is another instance of such a class.

In the next section, we will consider yet another way in which logical and symbolic artifacts can influence learning: we will see how such artifacts are useful to enable tractability, correctness, modularity and compositionality.

4 Logic for Machine Learning

There are two obvious ways in which a logical framework can provide insights on machine learning theory. First, consider that computational tractability is of central concern when applying logic in computer science, knowledge representation, database theory and search [65, 62, 71]. Thus, the natural question to wonder is whether these ideas would carry over to probabilistic machine learning. On the one hand, probabilistic extensions to tractable knowledge representation frameworks could be considered [57]. But on the other, as discussed previously, ideas from knowledge compilation, and the use of circuits, in particular, are proving very effective for designing tractable paradigms for machine learning. While there has always been an interest in capturing tractable distributions by means of low tree-width models [2], knowledge compilation has provided a way to also represent high tree-width models and enable exact inference for a range

of queries [82, 63]. See [24] for a comprehensive view on the use of knowledge compilation for machine learning.

The other obvious way logic can provide insights on machine learning theory is by offering a formal apparatus to reason about *context*. Machine learning problems are often positioned as atomic tasks, such as a classification task where regions of images need to be labeled as cats or dogs. However, even in that limited context, we imagine the resulting classification system as being deployed as part of a larger system, which includes various modules that communicate or interface with the classification system. We imagine an implicit accountability to the labelling task in that the detected object is either a cat or a dog, but not both. If there is information available that all the entities surrounding the object of interest have been labelled as lions, we would want to accord a high probability to the object being a cat, possibly a wild cat. There is a very low chance of the object being a dog, then. If this is part of a vision system on a robot, we should ensure that the robot never tramples on the object, regardless of whether it is a type of cat or a dog. To inspect such patterns, and provide meta-theory for machine learning, it can be shown that symbolic, programmatic and logical artifacts are enormously useful. We will specifically consider correctness, modularity and compositionality to explore the claim.

On the topic of correctness, the classical framework in computer science is *verification*: can we provide a formal specification of what is desired, and can the system be checked against that specification? In a machine learning context, we might ask whether the system, during or after training, satisfies a specification. The specification here might mean constraints about the physical laws of the domain, or notions of perturbation in the input space while ensuring that the labels do not change, or insisting that the prediction does not label an object as being both a cat and a dog, or otherwise ensuring that outcomes are not subject to, say, gender bias. Although there is a broad body of work on such issues, touching more generally on *trust* [86], we discuss approaches closer to the thrust of this article. For example, [49] show that a trained neural network can be verified by means of an SMT encoding of the network. In recent work, [96] show that the loss function of deep learning systems can be adjusted to logical constraints by insisting that the distribution on the predictions is proportional to the weighted model count of those constraints. In [63], prior (logical) constraints are compiled to a circuit to be used for probability estimation. In [80], circuits are shown to be amenable to training against probabilistic and causal prior constraints, including assertions about fairness, for example.

In [32, 67], a somewhat different approach to respecting domain constraints is taken: the low-level prediction is obtained as usual from a machine learning module, which is then interfaced with a probabilistic relational language and its symbolic engine. That is, the reasoning is positioned to be tackled directly by the symbolic engine. In a sense, such approaches cut across the three strands: the symbolic engine uses weighted model counting, the formulas in the language could be obtained by (say) entailment-based scoring, and the resulting language supports modularity and compositionality (discussed below).

While there is not much to be said about the distinction between finite vs infinite wrt correctness, many of these ideas are likely amenable to extensions to an infinite

setting in the ways discussed in the previous sections (e.g., considering constraints of a continuous or a countably infinite nature).

On the topic of modularity, recall that the general idea is to reduce, simplify or otherwise abstract a (probabilistic) computation as an atomic entity, which is then to be referenced in another, possibly more complex, entity. In standard programming languages, this might mean the compartmentalization and interrelation of computational entities. For machine learning, approaches such as probabilistic programming [40, 27] support probabilistic primitives in the language, with the intention of making learning modules re-usable and modular. It can be shown, for example, that the computational semantics of some of these languages reduce to WMC [36, 48]. Thus, in the infinite case, a corresponding reduction to WMI follows [31, 91, 1].

A second dimension to modularity is the notion of *abstraction*. Here, we seek to model, reason and explain the behavior of systems in a more tractable search space, by omitting irrelevant details. The idea is widely used in natural and social sciences. Think of understanding the political dynamics of elections by studying micro level phenomena (say, voter grievances in counties) versus macro level events (e.g., television advertisements, gerrymandering). In particular, in computer science, it is often understood as the process of mapping one representation onto a simpler representation by suppressing irrelevant information. In fact, integrating low-level behavior with high-level reasoning, exploiting relational representations to reduce the number of inference computations, and many other search space reduction techniques can all loosely be seen as instances of abstraction [8].

While there has been significant work on abstraction in deterministic systems [3], for machine learning, however, a probabilistic variant is clearly needed. In [47], an account of abstraction for loop-free propositional probabilistic programs is provided, where certain parts of the program (possibly involving continuous properties) can be reduced to a Bernoulli random variable. For example, suppose every occurrence of the continuous random variable x, drawn uniformly on the interval [0,1], in a program is either of the form $x \leq 7$ or of the form $x > 7$. Then, we could use a discrete random variable b with a 0.7 probability of being true to capture $x \leq 7$; and analogously, $\neg b$ to capture $x > 7$. The resulting program is likely to be simpler. In [8], an account of abstraction for probabilistic relational models is considered, where the notion of abstraction also extends to deterministic constraints and complex formulas. For example, a single probabilistic variable in the abstracted model could denote a complex logical formula in the original model. Moreover, the logical properties that enable verifying and inducing abstractions are also considered, and it is shown how WMC is sufficient for the computability of these properties (also see [48]).

Incidentally, abstraction brings to light a reduction between finite vs infinite: it is shown in [8] that the modelling of piecewise densities as weighted propositions, which is leveraged in WMI [12, 31], is a simple case of the more general account. Therefore, it is worthwhile to investigate whether this or other accounts of abstraction could emerge as general-purpose tools that allow us to inspect the conditions under which infinitary statements reduce to finite computations.

A broader point here is the role abstraction might play in generating explanations [44]. For example, a user's understanding of the domain is likely to be different from the

low-level data that a machine learning system interfaces with [92], and so, abstractions can capture these two levels in a formal way.

Finally, we turn to the topic of compositionality, which, of course, is closely related to modularity in that we want to distinct modules to come together to form a complex composition. Not surprisingly, this is of great concern in AI, as it is widely acknowledged that most AI systems will involve heterogeneous components, some of which may involve learning from data, and others reasoning, search and symbol manipulation [68]. In continuation with the above discussion, probabilistic programming is one such endeavor that purports to tackle this challenge by allowing modular components to be composed over programming and/or logical connectives [40, 27, 85, 67, 32, 76, 16, 11, 46, 5]. (See [71, 34, 64] for ideas in deterministic systems.) However, probabilistic programming only composes probabilistic computations, but does not offer an obvious means to capture other types of search-based computations, such as SAT, and integer and convex programming.

Recall that the computational semantics of probabilistic programs reduces to WMC [36, 48]. Following works such as [14, 33], an interesting observation made in [52] is that by appealing to a sum of products computation over different semiring structures, we can realize a large number of tasks such as satisfiability, unweighted model counting, sensitivity analysis, gradient computations, in addition to WMC. It was then shown in [9] that the idea could be generalized further for infinite domains: by defining a measure on first-order models, WMI and convex optimization can also be captured. As the underlying language is a logical one, composition can already be defined using logical connectives. But an additional, more involved, notion of composition is also proposed, where a sum of products over different semirings can be concatenated. To reiterate, the general idea behind these proposals [33, 52, 9] is to arrive at a principled paradigm that allows us to interface learned modules with other types of search and optimization computations for the compositional building of AI systems. See also [58] for analogous discussions, but where a different type of coupling for the underlying computations is suggested. Overall, we observed that a formal apparatus (symbolic, programmatic and logical artifacts) help us define such compositional constructions by providing a meta-theory.

5 Conclusions

In this article, we surveyed work that provides further evidence for the connections between logic and learning. Our narrative was structured in terms of three strands: logic versus learning, machine learning for logic, and logic for machine learning, but naturally, there was considerable overlap.

We covered a large body of work on what these connections look like, including, for example, pragmatic concerns such as the use of hard, domain-specific constraints and background knowledge, all of which considerably eases the requirement that all of the agent's knowledge should be derived from observations alone. (See discussions in [61] on the limitations of learned behavior, for example.) Where applicable, we placed an emphasis on how extensions to infinite domains are possible. In the very least, logical artifacts can help in constraining, simplifying and/or composing machine learning

entities, and in providing a principled way to study the underlying representational and computational issues.

In general, this type of work could help us move beyond the narrow focus of the current learning literature so as to deal with time, space, abstraction, causality, quantified generalizations, relational abstractions, unknown domains, unforeseen examples, among other things, in a principled fashion. In fact, what is being advocated is the tackling of problems that symbolic logic and machine learning might struggle to address individually. One could even think of the need for a recursive combination of strands 2 and 3: purely reactive components interact with purely cogitative elements, but then those reactive components are learned against domain constraints, and the cogitative elements are induced from data, and so on. More broadly, making progress towards a formal realization of *System 1* versus *System 2* processing might also contribute to our understanding of human intelligence, or at least capture human-like intelligence in automated systems.

References

1. A. Albarghouthi, L. D'Antoni, S. Drews, and A. V. Nori. Quantifying program bias. *CoRR*, abs/1702.05437, 2017.
2. F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2002.
3. B. Banihashemi, G. De Giacomo, and Y. Lespérance. Abstraction in situation calculus action theories. In *AAAI*, pages 1048–1055, 2017.
4. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, chapter 26, pages 825–885. IOS Press, 2009.
5. V. Belle. Logic meets probability: Towards explainable AI systems for uncertain worlds. In *IJCAI*, 2017.
6. V. Belle. Open-universe weighted model counting. In *AAAI*, pages 3701–3708, 2017.
7. V. Belle. Weighted model counting with function symbols. In *UAI*, 2017.
8. V. Belle. Abstracting probabilistic models: Relations, constraints and beyond. *Knowledge-Based Systems*, page 105976, 2020.
9. V. Belle and L. De Raedt. Semiring programming: A declarative framework for generalized sum product problems. *AAAI Workshop: Statistical Relational Artificial Intelligence*, 2020.
10. V. Belle and B. Juba. Implicitly learning to reason in first-order logic. In *Advances in Neural Information Processing Systems*, pages 3376–3386, 2019.
11. V. Belle and H. J. Levesque. Allegro: Belief-based programming in stochastic dynamical domains. In *IJCAI*, 2015.
12. V. Belle, A. Passerini, and G. Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *IJCAI*, pages 2770–2776, 2015.
13. M. Benedikt, K. Kersting, P. G. Kolaitis, and D. Neider. Logic and learning (dagstuhl seminar 19361). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020.
14. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint logic programming: syntax and semantics. *TOPLAS*, 23(1):1–29, 2001.
15. A. Bueff, S. Speichert, and V. Belle. Tractable querying and learning in hybrid domains via sum-product networks. *KR Workshop on Hybrid Reasoning*, 2018.
16. A. Bundy, K. Nuamah, and C. Lucas. Automated reasoning in the age of the internet. In *International Conference on Artificial Intelligence and Symbolic Computation*, pages 3–18. Springer, 2018.

17. R. Bunel, M. Hausknecht, J. Devlin, R. Singh, and P. Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. *arXiv preprint arXiv:1805.04276*, 2018.
18. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.
19. S. Chakraborty, D. J. Fremont, K. S. Meel, S. A. Seshia, and M. Y. Vardi. Distribution-aware sampling and weighted model counting for SAT. In *AAAI*, pages 1722–1730, 2014.
20. M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.
21. D. Chistikov, R. Dimitrova, and R. Majumdar. Approximate counting in SMT and value estimation for probabilistic programs. In *TACAS*, volume 9035, pages 320–334. 2015.
22. W. W. Cohen. PAC-learning nondeterminate clauses. In *AAAI*, pages 676–681, 1994.
23. A. Darwiche. New advances in compiling CNF to decomposable negation normal form. In *ECAI*, pages 328–332, 2004.
24. A. Darwiche. Three modern roles for logic in ai. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 229–243, 2020.
25. A. Darwiche and P. Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002.
26. L. De Raedt, A. Dries, I. Thon, G. Van den Broeck, and M. Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
27. L. De Raedt and A. Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.
28. L. De Raedt, R. Manhaeve, S. Dumancic, T. Demeester, and A. Kimmig. Neuro-symbolic= neural+ logical+ probabilistic. In *NeSy'19@ IJCAI, the 14th International Workshop on Neural-Symbolic Learning and Reasoning*, pages 1–4, 2019.
29. P. Dilkas and V. Belle. Generating random logic programs using constraint programming. *CoRR*, abs/2006.01889, 2020.
30. P. Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015.
31. P. Z. Dos Martires, A. Dries, and L. De Raedt. Exact and approximate weighted model integration with probability density functions using knowledge compilation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7825–7833, 2019.
32. A. Dries, A. Kimmig, J. Davis, V. Belle, and L. De Raedt. Solving probability problems in natural language. In *IJCAI*, 2017.
33. J. Eisner and N. W. Filardo. Dyna: Extending Datalog for modern AI. In *Datalog Reloaded*, volume 6702 of *LNCS*, pages 181–220. Springer, 2011.
34. A. Ensan and E. Ternovska. Modular systems with preferences. In *IJCAI*, pages 2940–2947, 2015.
35. R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.
36. D. Fierens, G. Van den Broeck, I. Thon, B. Gutmann, and L. De Raedt. Inference in probabilistic logic programs using weighted CNF's. In *UAI*, pages 211–220, 2011.
37. A. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *arXiv preprint arXiv:1905.06088*, 2019.
38. L. Getoor and B. Taskar, editors. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.
39. C. P. Gomes, A. Sabharwal, and B. Selman. Model counting. In *Handbook of Satisfiability*. IOS Press, 2009.

40. N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: A language for generative models. In *Proceedings of UAI*, pages 220–229, 2008.
41. M. Grohe and P. Lindner. Probabilistic databases with an infinite open-world assumption. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 17–31, 2019.
42. M. Grohe and M. Ritzert. Learning first-order definable concepts over structures of small degree. In *2017 32nd annual ACM/IEEE symposium on logic in computer science (LICS)*, pages 1–12. IEEE, 2017.
43. S. Gulwani. Dimensions in program synthesis. In *PPDP*, pages 13–24. ACM, 2010.
44. D. Gunning. Explainable artificial intelligence (xai). Technical report, DARPA/I20, 2016.
45. B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt. The magic of logical inference in probabilistic programming. *Theory and Practice of Logic Programming*, 11(4-5):663–680, 2011.
46. J. Y. Halpern. *Reasoning about Uncertainty*. MIT Press, 2003.
47. S. Holtzen, T. Millstein, and G. Van den Broeck. Probabilistic program abstractions. In *UAI*, 2017.
48. S. Holtzen, G. Van den Broeck, and T. Millstein. Dice: Compiling discrete probabilistic programs for scalable inference. *arXiv preprint arXiv:2005.09089*, 2020.
49. X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.
50. L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *I. J. Robotic Res.*, 32(9-10):1194–1227, 2013.
51. D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011.
52. A. Kimmig, G. Van den Broeck, and L. De Raedt. Algebraic model counting. *J. Appl. Log.*, 22:46–62, 2017.
53. S. Kolb, M. Mladenov, S. Sanner, V. Belle, and K. Kersting. Efficient symbolic integration for probabilistic inference. In *IJCAI*, 2018.
54. S. Kolb, P. Morettin, P. Zuidberg Dos Martires, F. Sommavilla, A. Passerini, R. Sebastiani, and L. De Raedt. The pywmi framework and toolbox for probabilistic inference using weighted model integration. *https://www. ijcai. org/proceedings/2019/*, 2019.
55. S. Kolb, S. Teso, A. Passerini, and L. De Raedt. Learning smt (lra) constraints using smt solvers. In *IJCAI*, pages 2333–2340, 2018.
56. D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques.* MIT Press, 2009.
57. D. Koller, A. Levy, and A. Pfeffer. P-classic: a tractable probablistic description logic. In *Proc. AAAI / IAAI*, pages 390–397, 1997.
58. P. Kordjamshidi, D. Roth, and K. Kersting. Systems ai: A declarative learning based programming perspective. In *IJCAI*, pages 5464–5471, 2018.
59. G. Lakemeyer and H. J. Levesque. Cognitive robotics. In *Handbook of Knowledge Representation*, pages 869–886. Elsevier, 2007.
60. L. Lamb, A. Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. *arXiv preprint arXiv:2003.00330*, 2020.
61. H. J. Levesque. *Common sense, the Turing test, and the quest for real AI.* MIT Press, 2017.
62. H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
63. Y. Liang, J. Bekker, and G. Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

64. Y. Lierler and M. Truszczynski. An abstract view on modularity in knowledge representation. In *AAAI*, pages 1532–1538, 2015.
65. Y. Liu and H. Levesque. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. IJCAI*, pages 522–527, 2005.
66. D. Lowd and P. Domingos. Learning arithmetic circuits. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 383–392, 2008.
67. R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pages 3749–3759, 2018.
68. G. Marcus and E. Davis. *Rebooting AI: Building artificial intelligence we can trust*. Pantheon, 2019.
69. D. Merrell, A. Albarghouthi, and L. D'Antoni. Weighted model integration with orthogonal transformations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.
70. B. Milch, B. Marthi, D. Sontag, S. J. Russell, D. L. Ong, and A. Kolobov. Approximate inference for infinite contingent bayesian networks. In *AISTATS*, pages 238–245, 2005.
71. D. G. Mitchell and E. Ternovska. A framework for representing and solving NP search problems. In *AAAI*, pages 430–435, 2005.
72. I. G. Mocanu, V. Belle, and B. Juba. Polynomial-time implicit learnability in smt. In *ECAI*, 2020.
73. A. Molina, A. Vergari, N. Di Mauro, S. Natarajan, F. Esposito, and K. Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *Thirty-second AAAI conference on artificial intelligence*, 2018.
74. P. Morettin, A. Passerini, and R. Sebastiani. Advanced smt techniques for weighted model integration. *Artificial Intelligence*, 275:1–27, 2019.
75. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.
76. D. Nitti, V. Belle, T. De Laet, and L. De Raedt. Planning in hybrid relational mdps. *Machine Learning*, 106(12):1905–1932, 2017.
77. D. Nitti, I. Ravkic, J. Davis, and L. D. Raedt. Learning the structure of dynamic hybrid relational models. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1283–1290. IOS Press, 2016.
78. F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *Proceedings of the VLDB Endowment*, 4(6):373–384, 2011.
79. F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.
80. I. Papantonis and V. Belle. On constraint definability in tractable probabilistic models. *arXiv preprint arXiv:2001.11349*, 2020.
81. D. Poole. First-order probabilistic inference. In *Proc. IJCAI*, pages 985–991, 2003.
82. H. Poon and P. Domingos. Sum-product networks: A new deep architecture. *UAI*, pages 337–346, 2011.
83. L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189, 2016.
84. J. Renkens, D. Shterionov, G. Van den Broeck, J. Vlasselaer, D. Fierens, W. Meert, G. Janssens, and L. De Raedt. ProbLog2: From probabilistic programming to statistical relational learning. In D. Roy, V. Mansinghka, and N. Goodman, editors, *Proceedings of the NIPS Probabilistic Programming Workshop,*, Dec. 2012. Accepted.
85. M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

86. C. Rudin and B. Ustun. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5):449–466, 2018.
87. S. J. Russell. Unifying logic and probability. *Commun. ACM*, 58(7):88–97, 2015.
88. S. Sanner and E. Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models. In *AAAI*, 2012.
89. P. Shenoy and J. West. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657, 2011.
90. P. Singla and P. M. Domingos. Markov logic in infinite domains. In *UAI*, pages 368–375, 2007.
91. S. Speichert and V. Belle. Learning probabilistic logic programs in continuous domains. In *ILP*, 2019.
92. S. Sreedharan, S. Srivastava, and S. Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *IJCAI*, pages 4829–4836, 2018.
93. D. Suciu, D. Olteanu, C. Ré, and C. Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.
94. L. G. Valiant. Robust logics. *Artificial Intelligence*, 117(2):231–253, 2000.
95. G. Van den Broeck. *Lifted Inference and Learning in Statistical Relational Models*. PhD thesis, KU Leuven, 2013.
96. J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning*, pages 5502–5511, 2018.
97. K. Xu, J. Li, M. Zhang, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.
98. R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018.
99. Z. Zeng and G. Van den Broeck. Efficient search-based weighted model integration. *arXiv preprint arXiv:1903.05334*, 2019.
100. P. Zuidberg Dos Martires, A. Dries, and L. De Raedt. Knowledge compilation with continuous random variables and its application in hybrid probabilistic logic programming. *arXiv preprint arXiv:1807.00614*, 2018.